# C++
Gyakorló feladatok
3. óra

A jegyzetet Umann Kristóf készítette. (2018. május 4.)

## 1. Láthatóság, élettartam, sztringkezelés, függvénytúlterhelés

**1.0.1. Feladat.** Legyen adott ez a C++ fájl:

**03_01str_main.cpp**

```
#include <iostream>

void insertSpace(const char *orig, char *dest) {
  // ...
}

int main() {
  const char *str = "Sziamizujs?";
  char buf[256];

  insertSpace(str, buf);

  std::cout << buf << std::endl;
}
```

Kimenet: `Szia mizujs?`

Valósítsuk meg az `insertSpace` függvényt, mely minden `'a'` karakter után beszúr egy space-t a paraméterként kapott strigbe! Feltételezhetjük, hogy `dest` legalább akkora tömbre mutat, mint `orig`.

Hogyan változna a megoldás, ha `char*` és `char[]` helyett `std::string` objektumokkal dolgoznánk?

**1.0.2. Feladat.** Egészítsünk ki egy korábbi feladatot!

**03_01static_main.cpp**

```
#include <iostream>

int findIf(/* ... */) {
  // ...
}

bool isSecondEven(int i) {
  // ...
}

int main() {
  int t[] = {1, 3, 4, 6, 7, 8, 10};

  std::cout << findIf(/* ... */, isSecondEven) << std::endl;
}
```

Kimenet: `3`

A korábban megvalósított `findIf` függvényhez írjunk egy újabb predikátumfüggvényt, mely egy statikus változó segítségével akkor ad igazat, ha a paraméterként kapott `int` a második páros szám!

Soroljunk fel legalább két indokot, miért célszerűtlen ez a megoldás valós alkalmazásra!

**1.0.3. Feladat.** Legyen adott ez a C++ fájl:

**03_01overload_main.cpp**

```cpp
#include <iostream>

struct IntWrapper {
  int i;
};

// *

int main() {
  int i = 10;
  IntWrapper wrap;
  wrap.i = 5;

  std::cout << add(i, wrap) << ' ' << add(wrap, i) << ' ' << add(wrap, wrap)
            << ' ' << add(i, i) << std::endl;
}
```

Kimenet: `15 15 10 20`

Valósítsuk meg az `add` függvényt, mely két tetszőleges szám összegét adja vissza! Függvénytúlterhelés segítségével érjük el azt, hogy ez a két paraméter mind `int` mind `IntWrapper` típusú is lehessen!

**1.0.4. Feladat.** Legyen adott ez a C++ fájl:

**03_02overload_main.cpp**

```cpp
#include <iostream>

struct IntWrapper {
  int i;
};

// *

int main() {
  IntWrapper wrap1, wrap2;
  wrap1.i = 10;
  wrap2.i = 5;

  std::cout << wrap1 * wrap2 << ' ' << wrap1 / wrap2 << ' ' << wrap1 + wrap2
            << ' ' << wrap1 - wrap2 << std::endl;
}
```

Kimenet: `50 2 15 5`

Operátor túlterheléssel valósítsuk meg a szorzás, osztás, összeadás és a kivonás műveletét! Minden operátor két `IntWrapper` típusú paramétert várjon, és a visszatérési értéke is egy `IntWrapper` típusú változó legyen! Gondoljuk meg, érték vagy referencia szerinti átvétel lenne-e hatékonyabb!

A kiíratáshoz terheljük túl a right shift operátort (`<<`) a jegyzetben található módon!